Small L^2 Error Is Not Enough: Function-Space Consistency for Neural Operators

Yuan Qiu, Peng Chen

{yqiu327, pchen402}@gatech.edu

Seeking Summer 2027 Internship

Physical AI / Scientific ML

Motivation

- Neural operators are popular machine-learning surrogates for the maps from parameters to solutions of partial differential equations.
- Mainstream paradigm for training: one samples the ground-truth solutions at a fixed set of nodes and then fits a network so that its pointwise outputs match those samples in a relative L^2 sense.
- We argue that this paradigm, although effective at producing predictions that look correct and report small relative L^2 error, is structurally insufficient for scientific use.

  > A small pointwise error on a fixed node set says almost nothing about the two properties that scientific workflows actually need: **off-node fidelity**, accuracy at points never seen during training, and **postprocessing consistency**, accuracy of derived quantities obtained by applying differential and integral operators to the predicted field.
- A function is not determined by its point values. Two functions can agree at every x_i yet differ arbitrarily between them and have wildly different derivatives.

  > Recovering a function from finitely many point values is well posed only after specifying an **interpolation rule** and **function space**, as finite element methods do explicitly through their spaces and **degrees of freedom**. However, pointwise neural operator training leaves it implicit, so the network supplies its own interpolation, tuned to minimize the mismatch at the nodes rather than to respect the numerical solver that generated the data.

One Test Case

Nonlinear diffusion-reaction equation on spatial domain $\Omega = (0,1)^2$

$$\sigma - p_1 \nabla u = 0, \qquad -\nabla\cdot\sigma + p_2 u^3 - p_3 = 0.$$

with boundary condition $u = 0$ on $\partial\Omega$. The task is to learn the parameter (functions)-to-solution (functions) mapping

$$p = (p_1, p_2, p_3) \mapsto s = (\sigma, u) = (\sigma_1, \sigma_2, u).$$

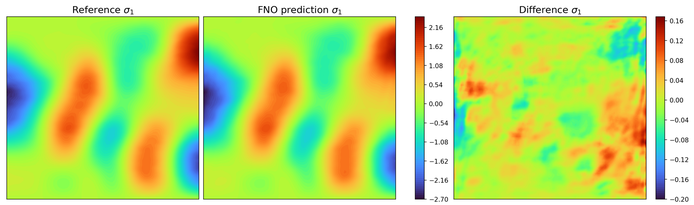
We assume a finite element discretization: 128×128 triangular mesh, $u \in \mathrm{CG}_2$ (continuous Lagrange element of degree 2), $\sigma \in \mathrm{RT}_1$ (Raviart-Thomas element of order 1, which conforms to $H(\mathrm{div})$).

- **Pointwise FNO[1]:** trained under the relative L^2 loss on pointwise evaluations over a uniform grid G,

$$\mathcal{L}(\theta) = \left\|\mathrm{FNO}_\theta(p|_G) - (\sigma,u)|_G\right\|_2 \Big/ \left\|(\sigma,u)|_G\right\|_2,$$

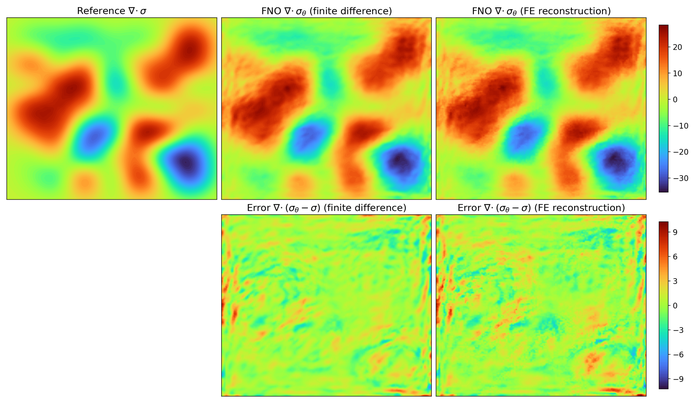
[1]Z. Li et al. Fourier neural operator for parametric partial differential equations. *ICLR* (2021).

The Flux Looks Correct



Reference, FNO prediction, and their difference for flux σ_1 on one test sample. The first two are almost indistinguishable. Similarly for σ_2.

But Its Divergence...



Top: the reference $\nabla\cdot\sigma$ is smooth, while the divergence of the FNO flux carries high-frequency contamination, whether computed by finite difference (middle) or by differentiating an $H(\mathrm{div})$ reconstruction (right). **Bottom:** the corresponding divergence errors.

We Compare Against...

- **Reconstruction baseline:** given **perfect** pointwise predictions (or $\mathrm{CG}_1 \times \mathrm{CG}_1$ representation), finite-element interpolate them into the $\mathrm{RT}_1 \times \mathrm{CG}_2$ space where original ground-truth solutions live.
- **Structure-aware[2]:** a reduced-basis neural operator (RBNO) predicting the solution in the $H(\mathrm{div})\times H^1$**-conforming space**, trained on the residual loss without labels,

$$\mathcal{J}(\sigma,u;p) = \|\sigma - p_1\nabla u\|_{L^2}^2 + \|-\nabla\cdot\sigma + p_2u^3 - p_3\|_{L^2}^2.$$

  where $\mathrm{DoFs}(\sigma,u) = \mathrm{DoFs}(\Phi_r)\cdot s_\theta(p|_G)$. Here Φ_r collects the $r = 512$ dominant POD modes, and s_θ the CNN-predicted coefficients.

[2]Y. Qiu, W. Dahmen & P. Chen. Variationally correct operator learning: Reduced basis neural operator with a posteriori error estimation. *Computer Methods in Applied Mechanics and Engineering* (2026).

Test Errors and Residual

	Pointwise FNO	Reconstr. baseline	Structure aware
Rel err in $L^2\times L^2$-norm	$\mathbf{3.4\times10^{-2}}$	$\mathbf{7.8\times10^{-4}}$	$\mathbf{4.5\times10^{-2}}$
Rel err in $H(\mathrm{div})\times H^1$-norm	1.1×10^{-1}	3.5×10^{-2}	3.5×10^{-2}
Conservation residual	$\mathbf{2.2}$	$\mathbf{2.4\times10^{-1}}$	$\mathbf{4.1\times10^{-2}}$
Rel err of Q_1 reaction integral	8.9×10^{-2}	8.8×10^{-4}	8.0×10^{-2}
Rel err of Q_2 right-edge flux	1.9×10^{-1}	1.5×10^{-4}	8.6×10^{-2}

Mean relative errors and residual on 480 test samples.

The conservation residual is given by

$$\|-\nabla\cdot\sigma + p_2u^3 - p_3\|_{L^2}^2,$$

and Q_1, Q_2 are the quantities of interest a practitioner might need:

$$Q_1 = \int_{\Omega_c} p_2u^3\,dx, \qquad Q_2 = \int_\Gamma \sigma\cdot n\,ds,$$

a reaction integral over the central box $\Omega_c = (0.25, 0.75)^2$, and the outward flux through the right edge of the boundary $\Gamma = \{1\}\times(0,1)$.

Where Should We Go Next?

- **Output spaces with the right structure.** Predict degrees of freedom in conforming function spaces rather than pointwise evaluations.

  > **Challenge:** harder to train, network architecture, optimizer, etc.
- **Losses that control postprocessing norms.** Training should penalize the error in a norm under which the postprocessing is stable.

  > **Challenge:** for a given downstream operator, identify the weakest norm that still guarantees a small postprocessing error, and make the corresponding loss efficiently computable.
- **Certified inference.** Neural operators should ship with ground-truth-free certificates, so that an individual prediction can be accepted, flagged for review, or corrected at deployment time.

  > **Challenge:** make the error estimates tight, rigorous, and tied to the quantities of interest.
- **Benchmarks that measure what matters.** Call for benchmarks that evaluate not only relative L^2 error, but also the **quantities relevant to downstream tasks** in scientific workflows.

This research is partially supported by NSF #2245111.